Privacy-Preserving Linear Regression on Distributed Data

Irene Giacomelli, Somesh Jha, and C. David Page

University of Wisconsin-Madison, Madison, WI, US

April 14, 2017

Abstract. Linear regression is an important statistical tool that models the relationship between some explanatory values and an outcome value using a linear function. In many current applications (*e.g.* predictive modelling in personalized healthcare), these values represent sensitive data owned by several different parties that are unwilling to share them. In this setting, training a linear regression model becomes challenging and needs specific cryptographic solutions. In this work, we propose a new system that can train two different variants of linear regression (*i.e.* ridge regression and lasso regression) on a dataset obtained by merging a finite number of private datasets. Our system assures that no extra information on a single private dataset is revealed to the entities performing the learning algorithm. Moreover, our solution is based on efficient cryptographic tools (*e.g.* Paillier's scheme and pseudorandom generator).

1 Introduction

Linear regression is an important statistical tool that models the relationship between some explanatory values (features) and an outcome value using a linear function. More precisely, given the data-points $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ where $\mathbf{x}_i$ is a vector of d real values (features) and y_i is a real value (outcome), a linear regression method is a learning algorithm for finding a vector $\mathbf{w}$ (the model) with d real components such that the value $\mathbf{w}(1)\mathbf{x}_i(1) + \cdots + \mathbf{w}(d)\mathbf{x}_i(d)$ is close to y_i for all $i = 1, \dots, n$. Despite its simple definition, a linear regression model is very useful. Indeed, $\mathbf{w}$ can be used to quantify the relationship between the features and the outcome (*e.g.* identify which features influence more directly the outcome value) and for future prediction (*e.g.* if a new vector of features with no known outcome is given, $\mathbf{w}$ can be used to make a prediction about it).

Motivation. In the standard statistics setting, it is assumed that the party performing the regression has direct access to all the data points in the training set in order to compute the model $\mathbf{w}$. This common assumption becomes non-trivial in some relevant areas where linear regression finds application (*e.g.* personalized medicine [C$^+$09]) because the data-points encode *sensitive* information owned by different and possibly mutually distrustful entities. Often, these entities will not (or can not) share their private data, making traditional linear regression algorithms difficult (or even impossible) to apply. On the other hand, it is known that having a large training dataset composed by a good variety of data-points (*e.g.* more relevant features or more data-points) improves the ability to compute a reliable model. Consider the following example: We would like to use a given linear regression method in order to predict the weight of a baby at birth on the basis of some ultrasound measurements made during last month of pregnancy (*e.g.* head circumference, femur length, etc). In order to avoid computing a biased model, we would like to run the selected learning algorithm on data points collected in different hospitals in different countries. On the other hand, each hospital legally cannot share (in the clear) patients' sensitive data (the measurements) with other hospitals or with a third party (*e.g.* a cloud-computing server). This real-life case exemplifies the challenge on which we focus in this work: training a linear regression model on data points that must be kept confidential and are owned by multiple parties.

This work. Our paper takes up this challenge and proposes an efficient solution in the setting in which the training set is a combination of data input by different parties (data-owners). Specifically, we consider the setting in which the features in the training dataset are distributed among different parties (*vertically-partitioned* dataset) and the setting in which each party has some of the data-points that form the training set (*horizontally-partitioned* dataset).
Our system is based on two efficiently implementable cryptographic primitives (public-key encryption schemes and pseudorandom generators) and is composed by two phases. In the first phase, we use a public-key encryption scheme with limited homomorphic property to let the data-owners securely submit their data to a third party. After this first step, the homomorphic property of the scheme is used to design an ad-hoc randomized algorithm that computes the desired linear regression model (*i.e.* ridge or lasso) from the encrypted version of the merged dataset. This algorithm is run by the party who collected the encrypted data with the help of a crypto-service provider. Neither one of these two parties have to handle the input data in the clear and moreover no extra information (beside that released by the model itself) is revealed to these two parties. To the best of our knowledge, our system is the first one to implement linear regression in the lasso version (*i.e.* with a norm-1 penalty applied to the model). Moreover, notice that the data-owners are active only in the first phase of the entire system. This makes our solution suitable for all applications in which the majority of data-owners are willing to help in order to run collaborative analysis but don't want to (or can not) spend to much resources for it.

Related work. The question of privacy-preserving machine learning was introduced in 2000 by two pioneering works [LP00,AS00]. Later on, privacy-preserving linear regression was considered in a number of different works (*e.g.* [KLSR04], [DHC04,SKLR04,KLSR05,KLSR09,HFN11,CDNN15,AHPW15]).
In 2013, Nikolaenko et al. introduced in [NWI$^+$13] the scenario considered in this paper: the privacy-preserving linear regression protocol has two phases. In the first phase there are possibly many data-owners that submit their private data to a third party. In the second phase, the third party is in charge of computing the

model with the help of a crypto-service provider. Their solution considers only the horizontally-partitioned setting for ridge regression and is based on additive encryption and garbled circuits (Yao's protocol). Specifically, in the second phase (model computing phase) of the protocol presented in [NWI$^+$13], the ridge regression model is computed using Yao's protocol to garble the circuit that solves a linear system of equations by computing the Cholesky decomposition of the associated coefficient matrix. Very recently, the system presented in [NWI$^+$13] is extended to the vertically-partitioned setting by the paper [GSB$^+$16]. Gascón et al. achieves this result using MPC techniques to allow the data-owners to compute shares of the merged dataset. Moreover, Gascón et al. also improves the running time of the second phase of the protocol presented in [NWI$^+$13] by designing a new conjugate gradient descent algorithm that is used in the place of Cholesky decomposition. Our paper follows this line of work and presents a new two-phase system. For the first phase, we extend the approach used by Nikolaenko et al. to the vertically-partitioned setting using an homomorphic encryption scheme that supports only one multiplication. For the second phase, we get rid of Yao's garbling scheme designing an ad-hoc two-party protocol that computes the desired regression model (ridge or lasso). This protocol exploit the homomorphic property of the underlying encryption scheme and employs pseudorandom generators.

Organization of the paper. In Section 2 we start our presentation recalling ridge and lasso linear regression. In Section 3 we describe the general framework of our system (*e.g.* parties involved, security assumptions, security definitions, etc.), and we also provide the definitions of the cryptographic primitives involved in the design of our system. Finally, in Section 4 we describes in detail all the protocols that form our two-phases system.

Standard notations. We use bold notation for vectors and capital letter for matrices (*e.g.* $\mathbf{x} \in \mathbb{R}^n$ is a column vector, $A \in \mathbb{R}^{n \times d}$ is matrix with n rows and d columns, both with real-value entries). We indicate with $\mathbf{x}(i)$ the i-th component of a vector $\mathbf{x}$ and with $A(i,j)$ the i-th component of the j-th column of the matrix A. The p-norm of a vector $\mathbf{x}$ is defined by $\|\mathbf{x}\|_p = \sqrt[p]{\sum_{i=1}^{n} |\mathbf{x}(i)|^p}$.

2 Linear Regression

A linear regression learning algorithm is a procedure that on input a dataset of n points $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ (where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$)[1] gives as output a vector $\mathbf{w}^* \in \mathbb{R}^d$ such that ${\mathbf{w}^*}^\top \mathbf{x}_i \approx y_i$ for all $i = 1, \ldots, n$. One common way to compute such a model $\mathbf{w}^*$ is to use the squared-loss function and the associated empirical error function (mean squared error):

$$f_{X,\mathbf{y}}(\mathbf{w}) = \|X\mathbf{w} - \mathbf{y}\|_2^2$$

[1] We assume that all the the values $\mathbf{x}_j(i)$ and y_j are rescaled to the same finite real interval (*e.g.* $[-1, 1]$).

where $X \in \mathbb{R}^{n \times d}$ is the matrix with the vector $\mathbf{x}_i^\top$ as ith row and $\mathbf{y} \in \mathbb{R}^n$ is the vector with the value y_i as ith component. Specifically, $\mathbf{w}^*$ is computed by minimizing a linear combination of the aforementioned error function and a regularization term,

$$\mathbf{w}^* \in \text{argmin}_{\mathbf{w}}\{f_{X,\mathbf{y}}(\mathbf{w}) + \lambda R(\mathbf{w})\}$$

The regularization term is added to avoid overfitting the training dataset and to compute simpler models. In practice, the most common regularization terms are the norm-2 ($R(\mathbf{w}) = \|\mathbf{w}\|_1$), which grantees a model with overall smaller components, and the norm-1 ($R(\mathbf{w}) = \|\mathbf{w}\|_1$), which provides a sparse model (*i.e.* a model with only few non-zero components). Note that, norm-1 regularization is often preferred since in practice it performs feature selection.
In the following, we briefly recall these two versions of regularized linear regression, known in the literature with the name of ridge and lasso regression, respectively. We always assume $\text{rk}(X) = d$.

Ridge Regression: The model $\mathbf{w}^*$ is computed by minimizing the function

$$\begin{aligned} F_{\text{ridge}}(\mathbf{w}) &= f_{X,\mathbf{y}}(\mathbf{w}) + \lambda f_{I,\mathbf{0}}(\mathbf{w}) \\ &= \|X\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2 \end{aligned}$$

where $\lambda \geq 0$ is fixed. Since F_{ridge} is a differentiable function, its minimum point[2] can be computed by solving the linear system corresponding to $\nabla F_{\text{ridge}}(\mathbf{w}) = \mathbf{0}$.

Lasso Regression: The model $\mathbf{w}^*$ is computed by minimizing the function

$$\begin{aligned} F_{\text{lasso}}(\mathbf{w}) &= f_{X,\mathbf{y}}(\mathbf{w}) + \lambda\|\mathbf{w}\|_1 \\ &= \|X\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1 \end{aligned}$$

where $\lambda \geq 0$ is fixed. Unlike ridge regression, lasso-penalized linear regression does not have a closed form solution, if $\lambda > 0$. Moreover, F_{lasso} is not differentiable everywhere, and finding its minimum point cannot even be accomplished by gradient descent. Nevertheless, it is possible to efficiently find an approximation of it. Indeed, it is known that finding the minimum point of F_{lasso} is equivalent to solving the following convex minimization problem [FT07]:

$$\begin{cases} \text{argmin}_{\mathbf{w} \in \mathbb{R}^d} f_{X,\mathbf{y}}(\mathbf{w}) \\ \text{subject to } \|\mathbf{w}\|_1 \leq s \end{cases} \tag{1}$$

where $s \in \mathbb{R}$ is fixed[3]. And an approximation of the solution[4] of (1) can be found using the *conditional gradient descent* method. This method, also known as the Frank-Wolfe (FW) algorithm (see Chapter 3 in [Bub15]), is an iterative

[2] If $\text{rk}(X) = d$, then the minimum point of F_{ridge} is unique.
[3] If we allow $s = +\infty$, then standard linear regression is also represented by (1).
[4] If $\text{rk}(X) = d$, then (1) has a unique solution.

algorithm that works in the following way: given an initial point $\mathbf{w}_1$ in $\mathbb{R}^d$ with $\|\mathbf{w}_1\|_1 \leq s$, then for any $j = 1, 2, \dots$ compute

$$\begin{cases} \mathbf{v}_j \in \text{argmin}_{\|\mathbf{v}\|_1 \leq s}\{\nabla f_{X,\mathbf{y}}(\mathbf{w}_j)^\top \mathbf{v}\} \\ \mathbf{w}_{j+1} = (1-\gamma_j)\mathbf{w}_j - \gamma_j \mathbf{v}_j \end{cases}$$

where $\gamma_j = \frac{2}{j+1}$. It is known that if there exists a positive constant M such that $||\mathbf{x}_i||_2 \leq M$ for all i, then it holds that

$$f_{X,\mathbf{y}}(\mathbf{w}_t) - \text{argmin}_{\mathbf{w}} F_{\text{lasso}}(\mathbf{w}) \leq \frac{8M^2}{t+1}$$

Finally, before describing in the next section the cryptographic setting we consider, we state here two mathematical facts about the mean squared error function that will be used in the following:

(F1) The function $f_{X,\mathbf{y}}$ is differentiable and $\nabla f_{X,\mathbf{y}}(\mathbf{w}) = 2X^\top(X\mathbf{w} - \mathbf{y})$.
From now on, we indicate $A = X^\top X$ ($d \times d$ matrix) and $\mathbf{b} = X^\top \mathbf{y}$ (vector of d components); note that since $\text{rk}(X) = d$, the matrix A is invertible.

(F2) Given a matrix $R \in \mathbb{R}^{d \times d}$ and a vector $\mathbf{r} \in \mathbb{R}^d$, then for any $\mathbf{w} \in \mathbb{R}^d$ we have that $f_{X,\mathbf{y}}(R\mathbf{w} - \mathbf{r}) = f_{XR,\mathbf{y}+X\mathbf{r}}(\mathbf{w})$.

3 The Privacy-Preserving Setting

We consider the setting where the training dataset is not available in the clear to the entity that wants to run the linear regression algorithm. Instead, the latter can access encrypted copies of the data and, for this reason, needs the help of the party handling the cryptographic keys in order to learn the desired model. More precisely, protocols in this paper are designed for the following parties (see Figure 3):

- The Data-Owners: there are m data-owners $\text{DO}_1, \dots, \text{DO}_m$; each data-owner DO_i has a private dataset $\mathcal{D}_i$ and is willing to share it only if encrypted.
- The Machine Learning Engine (MLE): this is the party that wants to run a linear regression algorithm on the dataset $\mathcal{D}$ obtained by merging the local datasets $\mathcal{D}_1, \dots, \mathcal{D}_m$, but has access only to the encrypted copies of them. For this reason, the MLE needs the help of the Crypto Service Provider;
- The Crypto Service Provider (CSP) takes care of initializing the encryption scheme used in the system and interacts with the MLE to help it in achieving its task (computing the linear regression model). The CSP manages the cryptographic keys and is the only entity capable of decrypting.

We assume that the MLE and the CSP do not collude and that all the parties involved are honest-but-curious. That is, they always follow the instruction of the protocol but try to learn extra information on the dataset from the messages

received during the execution of the protocol (*i.e.* passive security).
Moreover, we assume that for each pair of parties involved in the protocol there exists a private and authenticated peer-to-peer channel. In particular, communications between any two players cannot be eavesdropped[5].

Our goals are to ensure that the MLE obtains the regression model (the vector obtained by running the chosen regression algorithm on $\mathcal{D}$ in the clear) while both the MLE and the CSP learn any other information about the private datasets $\mathcal{D}_i$ beyond what is revealed by the model itself.

We achieve this goal by designing a *privacy-preserving system for linear regression*. The latter is a multi-party protocol run by the $m+2$ parties mentioned before and specified by a list of public parameters and a sequence of steps. The system described in this paper (Section 4) has the following two-phase form (see Figure 3):

– Phase 1 (merging the local datasets): the CSP generates key pair $(\mathbf{sk}, \mathbf{pk})$, stores $\mathbf{sk}$ and makes $\mathbf{pk}$ public; each DO_i sends to the MLE specific ciphertexts computed using $\mathbf{pk}$ and the values in $\mathcal{D}_i$.
– Phase 2 (computing the model): the MLE uses the ciphertexts received and private random values in order to obtain encryptions of new values that we call "masked data"; these encryptions are sent to the CSP; the latter decrypts and runs a given algorithm on the masked data. The output of this computation ("masked model") is a vector $\tilde{\mathbf{w}}$ that is sent back from the CSP to the MLE. In the last step of the protocol, the MLE computes $\tilde{\mathbf{w}}^*$ from $\tilde{\mathbf{w}}$.

Informally, we say that the system is *correct* if $\tilde{\mathbf{w}}^* \approx \mathbf{w}^*$ (that is, the model computed by the MLE is an approximation of the model computed by the learning algorithm in the clear). And we say that the system is *private* if the distribution of the masked data sent by the MLE to the CSP is independent from the distribution of the local inputs.

As we will see in section 4, the specific design of the protocol realising Phase 1 depends on the distributed setting: horizontally- or vertically-partitioned dataset, while the specific design of the protocol realising Phase 2 depends on the regularization version chosen for the linear regression: ridge or lasso.

3.1 Cryptographic Tools

To design our new privacy-preserving system for linear regression, we employ homomorphic encryption. An **additive encryption scheme** is defined by three algorithms:

1. the key-generation algorithm Gen takes in input the security parameter κ and outputs the pair of secret and public key, $\mathsf{Gen}(\kappa) \to (\mathbf{sk}, \mathbf{pk})$.

[5] Using proxy encryption, this requirements could be weakened. We leave this for future work

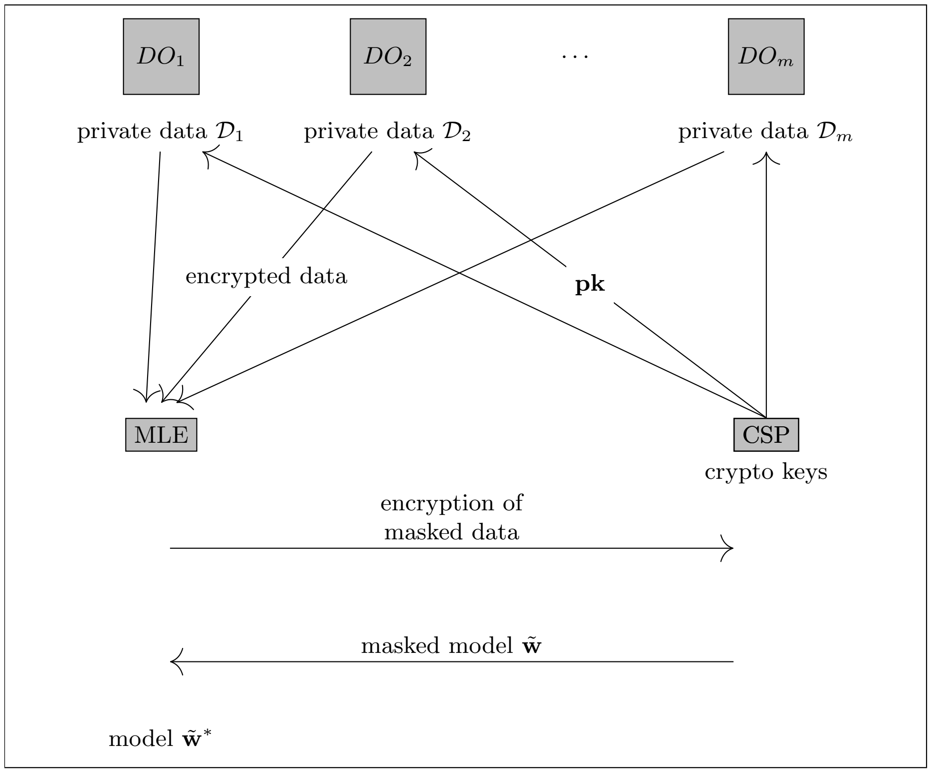
DO_1
DO_2
$\cdots$
DO_m
private data $\mathcal{D}_1$
private data $\mathcal{D}_2$
private data $\mathcal{D}_m$
encrypted data
pk
MLE
CSP
crypto keys
encryption of masked data
masked model $\tilde{\mathbf{w}}$
model $\tilde{\mathbf{w}}^*$

Fig. 1. Protocols setting overview.

2. the encryption algorithm Enc is a randomized algorithm that uses the public key to transform an element from $\mathcal{M}$ (finite plaintext space) in a ciphertext, $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{x}) \to \mathbf{c} \in \mathcal{C}$.
3. the decryption algorithm Dec is a deterministic function that takes in input a element from the ciphertext space $\mathcal{C}$ and the secret key and reveal the original plaintext (*i.e.* $\mathsf{Dec}_{\mathbf{sk}}(\mathbf{c}) = \mathbf{x}$).

The standard security property (semantic security) says that it is infeasible for any computationally bounded algorithm to gain extra information about a plaintext when given only its ciphertext and the public key **pk**. Moreover, we have the additive property: we assume that $(\mathcal{M}, +)$ is an additive group and that there exists an operation $\oplus$ on the ciphertext space $\mathcal{C}$ such that for any a-uple of ciphertexts $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{x}_1) \to \mathbf{c}_1, \ldots, \mathsf{Enc}_{\mathbf{pk}}(\mathbf{x}_a) \to \mathbf{c}_a$ (a integer), it holds that $\mathsf{Dec}_{\mathbf{sk}}(\mathbf{c}_1 \oplus \cdots \oplus \mathbf{c}_a) = \mathbf{x}_1 + \cdots + \mathbf{x}_a$. This implies that $\mathsf{Dec}_{\mathbf{sk}}(a\mathbf{c}) = a\mathbf{x}$, where $a\mathbf{c}^a$ means $\mathbf{c} \oplus \cdots \oplus \mathbf{c}$ a times.

Example 1. (Paillier's scheme.) Let $N = pq$ and $g \to \mathbb{Z}_{N^2}$, we have $\mathbf{pk} = (N, g)$ and $\mathcal{M} = \mathbb{Z}_N$. To encrypt M, sample r in $\mathbb{Z}_N^*$ and compute $c = g^M r^N \mod N^2$. In this case, $\oplus$ is the standard product of $\mathbb{Z}_{N^2}$, indeed:

- $c_1c_2 = (g^{M_1}r_1^N \mod N^2)(g^{M_2}r_2^N \mod N^2) = \mathsf{Enc}_{\mathbf{pk}}(M_1 + M_2)$
- $(c_1)^a = (g^{M_1}r_1^N \mod N^2)^a = \mathsf{Enc}_{\mathbf{pk}}(aM_1)$

A **one-time multiplicative encryption scheme** is an additive scheme that also support one multiplication among ciphertext. That is, we assume that in $(\mathcal{M}, +, \cdot)$ is a ring and that there is a product operation $\otimes$ defined on the ciphertext space $\mathcal{C}$ such that for any pair of ciphertexts $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{x}_1) \to \mathbf{c}_1$ and $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{x}_2) \to \mathbf{c}_2$, it holds that $\mathsf{Dec}_{\mathbf{sk}}(\mathbf{c}_1 \otimes \mathbf{c}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$.
An example of one-time multiplicative scheme can be found in [GHV10][6].

Example 2. (GHV scheme [GHV10].) Let N, M be integers and p, q primes ($M, q = poly(N)$). We have $\mathcal{M} = \mathbb{Z}_p^{M\times M}$ and

1. $\mathsf{Gen}(N)$: run the trapdoor function in [AP09] and obtain a matrix $A \in \mathbb{Z}_q^{M\times N}$ and an invertible matrix $T \in \mathbb{Z}^{M\times N}$ such that $TA = 0 \mod q$. Define $\mathbf{pk} = A$ and $\mathbf{sk} = T$;
2. $\mathsf{Enc}_{\mathbf{pk}}(M)$: sample S, X and compute $C = AS + pX + M \mod q$;
3. $\mathsf{Dec}_{\mathbf{sk}}(C)$: let $E = TCT^\top \mod q$ and output $T^{-1}E(T^\top)^{-1} \mod p$.

4 Privacy-Preserving Linear Regression

In this section, we describe our two-phases system for training a regression model (with norm-2 or norm-1 regularization term) in the setting described in Section 3. Specifically, we describe how to implement Phase 1 (merging the datasets) and Phase 2 (computing the model) in Section 4.1 and 4.2, respectively.

Before, describing in the detail the protocols implementing this idea, we fix here the notation and the assumptions we have about data representation.

Data representation: We assume that the entries of X and $\mathbf{y}$ are real number from a bounded interval $I = [-10^r, 10^r]$ and have at most ℓ decimal digit. As in previous work [NWI+13,CDNN15,GSB+16], the usage of cryptographic tools implies that these real values are represented by *fixed-point data type.* We call fixed-point representation with resolutions ℓ of the real number a the integer $[a] = \lfloor a \cdot 10^\ell \rfloor$. Since we assume that all the entries stay in the interval I, there exists a positive integer k such that the fixed-point representations of the aforementioned values stay in the set $\mathbb{Z}_{<\ell+r>} = \{x \in \mathbb{Z} \mid -10^{\ell+r} < x < 10^{\ell+r}\}$. This standard method to represent real values is not enough in the setting considered in this paper. Indeed, in order to prove the privacy property of the protocols designed in the following, we rely on the fact that some values are sampled from a finite ring. For this reason, we need to map the set $\mathbb{Z}_{<\ell+r>}$ to the ring $\mathbb{Z}_q$ for some large q ($q > 10^{\ell+r}$). For this, we use the map $x \mapsto x \mod q$. If no overflow occurred (q large enough), we can compute on fixed-point

[6] Similar to the Boneh-Goh-Nassim cryptosystem [BGN05], but with a more efficient encryption and decryption algorithms.

numbers from $\mathbb{Z}_{<\ell+r>}$ using the arithmetic of $\mathbb{Z}_q$ for a big enough q. In this way, we obtain a two-steps map from $\mathbb{R}$ to $\mathbb{Z}_q$ that can also be easily inverted:

$$\begin{aligned} emb : \mathbb{R} &\longrightarrow \mathbb{Z}_{<\ell+r>} \longrightarrow \mathbb{Z}_q \\ x &\mapsto \lfloor x10^{\ell} \rceil \mapsto \lfloor x10^{\ell} \rceil \mod q \\ emb^{-1} : \mathbb{Z}_q &\longrightarrow \mathbb{Z}_{<\ell+r>} \longrightarrow \mathbb{R} \\ y &\mapsto z \mapsto z10^{-\ell} \end{aligned}$$

with $z = \begin{cases} y & \text{if } 0 \le y \le q/2 \\ y - q & \text{if } q/2 < qy \le q-1 \end{cases}$

From now on, we assume that the map emb is used to represent all input values and regression parameters (λ or s) as element in $\mathbb{Z}_q$.

4.1 Phase 1: merging the dataset

Assume that the dataset represented by the matrix X is horizontally-portioned in m datasets. This means that each DO_i holds some rows of the matrix X. We assume that the correspondence between rows and parties is publicly known and has this form: the data-owner DO_i holds $\mathcal{D}_i$ and

$$\mathcal{D}_i = \{(\mathbf{x}_{n_{i-1}+1}, y_{n_{i-1}+1}), \dots, (\mathbf{x}_{n_i}, y_{n_i})\}$$

for $i = 1, \dots, m$ ($n_0 = 1, n_m = n$). In this case, as already noticed in [NWI+13], we have the following. Recall that $A = X^\top X$, $\mathbf{b} = X^\top \mathbf{y}$ and define $A_i = \mathbf{x}_i \mathbf{x}_i^\top$, $\mathbf{b}_i = y_i \mathbf{x}_i$ for $i = 1, \dots n$, then

$$A = \sum_{i=1}^{n} A_i \quad \text{and} \quad \mathbf{b} = \sum_{i=1}^{n} \mathbf{b}_i.$$

Each A_i and $\mathbf{b}_i$ can be computed locally by the party DO_i; then, if additively homomorphic encryptions of such values are sent to the MLE, the latter can easily compute encryptions of A and $\mathbf{b}$. This is captured in protocol Π^1_{hor} (Figure 2).

On the other hand, in the vertically-partitioned setting we assume that each DO_i holds some columns of X. We assume the correspondence between columns and parties is publicly known and can be represented in the following way. For $i = 1, \dots, m-1$ ($n_0 = 1, n_{m-1} = d$), define

$$X_i = \begin{pmatrix} \mathbf{x}_1(n_{i-1}+1) \dots \mathbf{x}_1(n_i) \\ \vdots \qquad\qquad \vdots \\ \mathbf{x}_n(n_{i-1}+1 \dots \mathbf{x}_n(n_i) \end{pmatrix} \quad \text{and} \quad X_m = \mathbf{y}$$

and let $\mathcal{D}_i = \{X_i\}$. The data-owner DO_i holds $\mathcal{D}_i$. In this case, we have that

$$A = \begin{pmatrix} X_1^\top X_1 & X_1^\top X_2 & \dots & X_1^\top X_{m-1} \\ \vdots & \vdots & & \vdots \\ X_{m-1}^\top X_1 & X_{m-1}^\top X_2 & \dots & X_{m-1}^\top X_{m-1} \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} X_1^\top X_m \\ \vdots \\ X_{m-1}^\top X_m \end{pmatrix} \qquad (2)$$

Protocol Π^1_{hor}

- Parties: CSP and MLE with no input
  DO$_i$ with input $\mathcal{D}_i$ for all $i = 1, \ldots, m$.
- Output: MLE gets A' and $\mathbf{b}'$ encryptions of A and $\mathbf{b}$, respectively.
- Public parameters: number of features d, total numbers of data points d.

Assume that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is an additive encryption scheme with plaintext space $\mathcal{M}$ (and $\mathbb{Z}_q \subseteq \mathcal{M}$) and security parameter κ.

Step 1: the CSP runs $\mathsf{Gen}(\kappa) \to (\mathbf{sk}, \mathbf{pk})$ and makes $\mathbf{pk}$ public, while it keeps $\mathbf{sk}$ secret.

Step 2: for all $k = 1, \ldots, m$, DO$_k$ does the following

a) (local computation) for all $i = n_{k-1} + 1, \ldots, n_k$, compute $A_i = \mathbf{x}_i \mathbf{x}_i^\top$ and $\mathbf{b}_i = y_i \mathbf{x}_i$;

b) (encryption) for all $i = n_{k-1} + 1, \ldots, n_k$, compute[a]

$$A'_i(k, j) = \mathsf{Enc}_{\mathbf{pk}}(A_i(k, j))$$
$$\mathbf{b}'_i(j) = \mathsf{Enc}_{\mathbf{pk}}(\mathbf{b}_i(j));$$

for all $k, j = 1, \ldots, d$;

c) for all $i = n_{k-1} + 1, \ldots, n_k$, send A'_i and $\mathbf{b}'_i$ to the MLE.

Step 3: the MLE does the following

a) (datasets merging) compute the values

$$A'(i, j) = \bigoplus_{k=1}^{n} A'_k(i, j)$$
$$\mathbf{b}'(i) = \bigoplus_{k=1}^{n} \mathbf{b}'_k(i)$$

for all $i, j = 1, \ldots, d$ and $j \geq i$;

[a] Batching techniques can be used to improve efficiency of the practical implementation.

Fig. 2. The protocol Π^1_{hor} implements Phase 1 in the horizontally-distributed setting.

If all values in X are encrypted, computing an encryption of $X_i^\top X_j$ requires only depth-one multiplication of ciphertexts. Thus an encryption of A and $\mathbf{b}$ can be computed from encryptions of the local datasets, if the underling scheme is one-time multiplicative. This is what protocol Π^1_{ver} (Figure 3) describes.

Protocol $\varPi^1_{\mathbf{ver}}$

– Parties: CSP and MLE with no input
  DO$_i$ with input $\mathcal{D}_i$ for all $i = 1, \ldots, m$.
– Output: MLE gets A' and $\mathbf{b}'$ encryptions of A and $\mathbf{b}$, respectively.
– Public parameters: number of features d, total numbers of data points n.

Assume that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a one-time multiplicative encryption scheme with plaintext space $\mathcal{M}$ (and $\mathbb{Z}_q \subseteq \mathcal{M}$) and security parameter κ.

Step 1: the CSP runs $\mathsf{Gen}(\kappa) \to (\mathbf{sk}, \mathbf{pk})$ and makes $\mathbf{pk}$ public, while it keeps $\mathbf{sk}$ secret.

Step 2: for all $i = 1, \ldots, m$, DO$_i$ does the following
  a) (encryption) compute[a] $X'_i(\ell, j) = \mathsf{Enc}_{\mathbf{pk}}(X_i(\ell, j))$ for all $\ell = 1, \ldots, n$ and $j = 1, \ldots, (n_i - n_{i-1})$;

  b) send X'_i to the MLE.

Step 3: (datasets merging) the MLE does computes all the encryptions of $X_i^\top X_j$ with $j \geq i$ and $i, j = 1, \ldots m$ as

$$\mathsf{Enc}_{\mathbf{pk}}(X_i^\top X_j(\ell, k)) = \bigoplus_{t=1}^{n} X'_i(t, \ell) \otimes X'_j(t, k)$$

and uses this values to forms encryptions of A and $\mathbf{b}$ according to (2).

[a] Batching techniques can be used to improve efficiency of the practical implementation.

Fig. 3. The protocol $\varPi^1_{\mathrm{ver}}$ implements Phase 1 in the vertically-distributed setting.

4.2 Phase 2: computing the model

Before, showing how to implement Phase 2, we state here the lemma that we will use in the following to prove privacy. Let $(\mathbb{Z}_q)^{k\times k}_{\mathrm{inv}} = \{M \in \mathbb{Z}_q^{k\times k} \mid M \text{ invertible } \}$.

Lemma 1. *Fix $M \in (\mathbb{Z}_q)^{k\times k}_{\mathrm{inv}}$ and $\mathbf{v} \in \mathbb{Z}_q^k$; assume that R is sampled uniformly at random from $(\mathbb{Z}_q)^{k\times k}_{\mathrm{inv}}$ and $\mathbf{r}$ is sampled uniformly at random from $\mathbb{Z}_q^k$. Then for any fixed $\mathbf{v} \in \mathbb{Z}_q^d$, the distribution of $(MR, \mathbf{v} + M\mathbf{r},)$ is the uniform one over $(\mathbb{Z}_q)^{k\times k}_{\mathrm{inv}} \times \mathbb{Z}_q^k$.*

Protocol $\boldsymbol{\varPi^2_{\mathrm{ridge}}}$

Recall that in ridge regression we compute the model $\mathbf{w}^*$ by finding the minimum of the function $F_{\mathrm{ridge}}(\mathbf{w})$ (Section 2). Now, consider the masked version of it given by:

$$\tilde{F}_{\text{ridge}}(\mathbf{w}) = f_{XR,\mathbf{y}+X\mathbf{r}}(\mathbf{w}) + \lambda f_{R,\mathbf{r}}(\mathbf{w})$$

where R is any $d \times d$ matrix and $\mathbf{r}$ is a vector of d components.

Lemma 2. *If R is invertible, $\tilde{\mathbf{w}} = \text{argmin}_{\mathbf{w}} \tilde{F}_{\text{ridge}}(\mathbf{w})$ and $\mathbf{w}^* = R\tilde{\mathbf{w}} - \mathbf{r}$, then $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} F_{\text{ridge}}(\mathbf{w})$.*

Proof. It follows from (F2) that, for any $\mathbf{w}$,

$$F_{\text{ridge}}(\mathbf{w}) = \tilde{F}_{\text{ridge}}(R^{-1}(\mathbf{w}+\mathbf{r})) \geq \tilde{F}_{\text{ridge}}(\tilde{\mathbf{w}}) = F_{\text{ridge}}(\mathbf{w}^*)$$

Lemma 2 implies that the problem of computing $\mathbf{w}^*$ can be reduced to the problem of finding the minimum point of the masked function $\tilde{F}_{\text{ridge}}$. Since the latter function is differentiable, $\tilde{\mathbf{w}}$ can be computed as solution of $\nabla \tilde{F}_{\text{ridge}}(\mathbf{w}) = \mathbf{0}$. A simple computation shows that

$$\begin{aligned} \nabla \tilde{F}_{\text{ridge}}(\mathbf{w}) &= 2[(XR)^\top (XR\mathbf{w} - \mathbf{y} - X\mathbf{r}) + \lambda R^\top (R\mathbf{w} - \mathbf{r})] \\ &= 2R^\top [(A+\lambda I)R\mathbf{w} - (\mathbf{b} + (A+\lambda I)\mathbf{r})] \end{aligned}$$

It follows that $\tilde{\mathbf{w}} = [(A+\lambda I)R]^{-1}(\mathbf{b} + (A+\lambda I)\mathbf{r})$.
Given this, we design the protocol Π^2_{ridge} (Figure 4). In this protocol encryptions of $C = (A+\lambda I)R$ and $\mathbf{d} = \mathbf{b} + (A+\lambda I)\mathbf{r}$ are computed by the MLE (who has access to the encryption of A and $\mathbf{b}$) and sent by it to the CSP. The latter decrypts and computes $\tilde{\mathbf{w}}$. Notice that we require that MLE samples R uniformly at random from all invertible $d \times d$ matrices with coefficient in $\mathbb{Z}_q$, and $\mathbf{r}$ uniformly at random from $\mathbb{Z}_q^d$. In this way, the privacy property follows from Lemma 1 (take $M = A + \lambda I$ and $\mathbf{v} = \mathbf{b}$): the joint distribution of the matrix C and the vector $\mathbf{d}$ is independent from the input data-points. In the last step, the MLE receives $\tilde{\mathbf{w}}$ and computes $R\tilde{\mathbf{w}} - \mathbf{r}$. The correctness of Π^2_{ridge} follows from Lemma 2.

Protocol Π^2_{lasso}

Recall that in lasso regression we compute the model $\mathbf{w}^*$ by finding an approximation of the solution of problem (1). Now, consider the following masked version of (1):

$$\begin{cases} \text{argmin}_{\mathbf{w} \in \mathbb{R}^d} f_{XR,\mathbf{y}+X\mathbf{r}}(\mathbf{w}) \\ \text{subject to } \|R\mathbf{w} - \mathbf{r}\|_1 \leq s \end{cases} \tag{3}$$

where R is any $d \times d$ matrix and $\mathbf{r}$ is a vector of d components.

Lemma 3. *Let $\tilde{\mathbf{w}}$ be solution of (3). If R is invertible and $\hat{\mathbf{w}} = R\tilde{\mathbf{w}} - \mathbf{r}$, then $\hat{\mathbf{w}}$ is solution of (1). Moreover, if $\tilde{\mathbf{w}}_t$ is such that $f_{XR,\mathbf{y}+X\mathbf{r}}(\tilde{\mathbf{w}}_t) - f_{XR,\mathbf{y}+X\mathbf{r}}(\tilde{\mathbf{w}}) \leq \epsilon$ and $\mathbf{w}_t = R\tilde{\mathbf{w}}_t - \mathbf{r}$, then $f_{X,\mathbf{y}}(\mathbf{w}_t) - f_{X,\mathbf{y}}(\hat{\mathbf{w}}) \leq \epsilon$.*

Proof. It follows from (F2) that, for any $\mathbf{u}$ with $\|\mathbf{u}\|_1 \leq s$ we have

$$f_{X,\mathbf{y}}(\mathbf{u}) = f_{X,\mathbf{y}}(R\mathbf{w} - \mathbf{r}) = f_{XR,\mathbf{y}+X\mathbf{r}}(\mathbf{w}) \geq f_{XR,\mathbf{y}+X\mathbf{r}}(\tilde{\mathbf{w}}) = f_{X,\mathbf{y}}(\hat{\mathbf{w}})$$

Moreover, $\|\hat{\mathbf{w}}\|_1 = \|R\tilde{\mathbf{w}} - \mathbf{r}\|_1 \leq s$

Protocol $\Pi^2_{\textbf{ridge}}$

The protocol Π^1_{hor} or the protocol Π^1_{ver} has been previously run.

- Parties: CSP knows $\mathbf{sk}$, the MLE knows A' and $\mathbf{b}'$
- Output: MLE gets $\tilde{\mathbf{w}}^*$
- Public parameters: number of features d, total numbers of data points n, penalty parameter λ.

Step 1: the MLE does the following

a) compute the values

$$M'(i,j) = \begin{cases} A'(i,j) \oplus \mathsf{Enc}_{\mathbf{pk}}(\lambda) & \text{if } i = j \\ A'(i,j) & \text{otherwise} \end{cases}$$

for all $i, j = 1, \dots, d$;

b) (sampling) sample[a] $R \leftarrow (\mathbb{Z}_q)^{d \times d}_{\text{inv}}$ and $\mathbf{r} \leftarrow \mathbb{Z}_q^d$;

c) (masking) compute[b] the values

$$C'(i,j) = \bigoplus_{k=1}^{d} R(k,j) M'(i,k)$$

$$\mathbf{d}'(i) = \mathbf{b}'(i) \oplus \left(\bigoplus_{k=1}^{d} \mathbf{r}(k) M'(i,k) \right)$$

for all $i, j = 1, \dots, d$;

d) send the matrix C' and the vector $\mathbf{d}'$ to the CSP.

Step 2: the CSP does the following

a) compute the values

$$C(i,j) = \mathsf{Dec}_{\mathbf{sk}}(C'(i,j))$$
$$\mathbf{d}(i) = \mathsf{Dec}_{\mathbf{sk}}(\mathbf{d}'(i))$$

for all $i, j = 1, \dots, d$;

b) compute $\tilde{\mathbf{w}} = C^{-1}\mathbf{d}$;

c) send to the MLE the vector $\tilde{\mathbf{w}}$.

Step 3: the MLE computes $\tilde{\mathbf{w}}^* = R\tilde{\mathbf{w}} - \mathbf{r}$.

[a] This can be implemented using efficient PRG [VZ12] and rejection sampling.
[b] Note that $R(k,j)M'(i,k)$ means $M'(i,k) \oplus \cdots \oplus M'(i,k)$ with $R(k,j)$ addends.

Fig. 4. The protocol Π^2_{ridge} implements Phase 2 of our system when a norm-2 penalization is chosen in order to compute the regression model.

From Lemma 3, it is clear that computing $\tilde{\mathbf{w}}_t$ that is an approximation of the solution of (3) is sufficient in order to compute the desired model $\mathbf{w}^*$. If we use the FW algorithm to compute $\tilde{\mathbf{w}}_t$, we need to solve $t-1$ problems of this form

$$\begin{cases} \text{argmin}_{\mathbf{v}\in\mathbb{R}^d} \nabla f_{XR,\mathbf{y}+X\mathbf{r}}(\tilde{\mathbf{w}}_j)^\top \mathbf{v} \\ \text{subject to } \|R\mathbf{v}-\mathbf{r}\|_1 \le s \end{cases}$$

Since R is invertible, we can replace $R\mathbf{v}-\mathbf{r}$ with $R\mathbf{u}$ and we obtain

$$\begin{cases} \text{argmin}_{\mathbf{v}\in\mathbb{R}^d} \nabla f_{XR,\mathbf{y}+X\mathbf{r}}(\tilde{\mathbf{w}}_{t-1})^\top \mathbf{u} \\ \text{subject to } \|R\mathbf{u}\|_1 \le s \end{cases} \tag{4}$$

Using (F1), we have:

$$\nabla f_{XR,\mathbf{y}+X\mathbf{r}}(\mathbf{u}) = 2(XR)^\top(XR\mathbf{w}-\mathbf{y}-X\mathbf{r}) = 2R^\top[AR\mathbf{w}-(\mathbf{b}+A\mathbf{r})]$$

Therefore, (4) becomes

$$\begin{cases} \text{argmin}_{\mathbf{v}\in\mathbb{R}^d} [AR\tilde{\mathbf{w}}_{t-1}-(\mathbf{b}+A\mathbf{r})]^\top R\mathbf{v} \\ \text{subject to } \|R\mathbf{v}\|_1 \le s \end{cases}$$

which is equivalent to

$$\begin{cases} \text{argmin}_{\mathbf{v}\in\mathbb{R}^d} [AR\tilde{\mathbf{w}}_{t-1}-(\mathbf{b}+A\mathbf{r})]^\top \mathbf{v} \\ \text{subject to } \|\mathbf{v}\|_1 \le s \end{cases} \tag{5}$$

Based on this, we design the protocol Π^2_{lasso} (Figure 5) that realizes Phase 2 of our system when a norm-1 penalty is chosen. In this protocol, MLE computes encryptions of $S = AR$ and $\mathbf{s} = \mathbf{b} + A\mathbf{r}$ where R is sampled uniformly at random from all invertible $d \times d$ matrices and $\mathbf{r}$ is an uniformly at random sampled vector of d components. The encryptions of S and $\mathbf{s}$ are sent to the CSP, who decrypts the values received and computes the vector $\tilde{\mathbf{w}}_t$ via the FW algorithm (using (5) for each iteration). This vector is sent back to the MLE. Finally, the MLE computes $\tilde{\mathbf{w}}^*$ as $R\tilde{\mathbf{w}}_t - \mathbf{r}$. The correctness of protocol Π^2_{lasso} follows from Lemma 3. The privacy property follows from Lemma 1 where $M = A$ and $\mathbf{v} = \mathbf{b}$.

4.3 Security proof

To formally prove security, we use the standard simulation-based definition [Gol01]. Consider a public function $\phi : (\{0,1\}^k)^n \to \{0,1\}^\ell$ and let $P_1, \ldots, P_n$ be n players modelled as PPT machines. Each player P_i holds the value $\mathbf{a}_i \in \{0,1\}^k$ and wants to compute the value $\phi(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ while keeping his input private. The players can communicate among them using point-to-point secure channels in the synchronous model. If necessary, we also allow the players to use a broadcast channel. To achieve their goal, the players jointly run a n-party MPC

Protocol Π^2_{lasso}

The protocol Π^1_{hor} or the protocol Π^1_{ver} has been previously run.

- Parties: CSP knows $\mathbf{sk}$, the MLE knows A' and $\mathbf{b}'$
- Output: MLE gets $\tilde{\mathbf{w}}^*$
- Public parameters: number of features d, total numbers of data points n, penalty parameter s, number of steps t, initial point $\mathbf{w}_1$.

Step 1: the MLE does the following

a) (sampling) sample[a] $R \leftarrow (\mathbb{Z}_q)^{d\times d}_{\text{inv}}$ and $\mathbf{r} \leftarrow \mathbb{Z}_q^d$;

b) (masking) compute[b] the values

$$S'(i,j) = \bigoplus_{k=1}^{d} R(k,j)A'(i,k)$$

$$\mathbf{s}'(i) = \mathbf{b}'(i) \oplus \left(\bigoplus_{k=1}^{d} \mathbf{r}(k)A'(i,k)\right)$$

for all $i, j = 1, \ldots, d$;

c) send the matrix S' and the vector $\mathbf{s}'$ to the CSP.

Step 2: the CSP does the following

a) compute the values

$$S(i,j) = \mathsf{Dec}_{\mathbf{sk}}(S'(i,j))$$
$$\mathbf{s}(i) = \mathsf{Dec}_{\mathbf{sk}}(\mathbf{s}'(i))$$

for all $i, j = 1, \ldots, d$;

b) compute $\tilde{\mathbf{w}}_t$ using the FW algorithm. That is, take $\tilde{\mathbf{w}}_1 = \mathbf{w}_1$ and for any $j = 1, 2, \ldots, t-1$ compute

$$\begin{cases} \mathbf{v}_j \in \operatorname{argmin}_{\|\mathbf{v}\|_1 \le s}\{(S\tilde{\mathbf{w}}_j - \mathbf{s})^\top \mathbf{v}\} \\ \tilde{\mathbf{w}}_{j+1} = (1-\gamma_j)\tilde{\mathbf{w}}_j - \gamma_j \mathbf{v}_j \end{cases}$$

c) send to the MLE the vector $\tilde{\mathbf{w}}_t$.

Step 3: the MLE computes $\tilde{\mathbf{w}}^* = R\tilde{\mathbf{w}}_t - \mathbf{r}$.

[a] This can be implemented using efficient PRG [VZ12] and rejection sampling.
[b] Note that $R(k,j)A'(i,k)$ means $M'(i,k) \oplus \cdots \oplus A'(i,k)$ with $R(k,j)$ addends.

Fig. 5. The protocol Π^2_{lasso} implements Phase 2 of our system when a norm-1 penalization is chosen in order to compute the regression model.

protocol Π. The latter is a protocol for n players that is specified via the next-message functions: there are several rounds of communication and in each round the player P_i sends to other players a message that is computed as a deterministic function of the internal state of P_i (his initial input $\mathbf{a}_i$ and his random tape $\mathbf{k}_i$) and the messages that P_i has received in the previous rounds of communications. The *view* of the player P_j, denoted by $\mathrm{View}_{P_j}(\mathbf{a}_1, \ldots, \mathbf{a}_n)$, is defined as the concatenation of the private input $\mathbf{a}_j$, the random tape $\mathbf{k}_j$ and all the messages received by P_j during the execution of Π. Finally, the output of Π for the player P_j can be computed from the view View_{P_j}. In order to be private, the protocol Π needs to be designed in such a way that a curious player P_i can not infer information about $\mathbf{a}_j$ with $j \neq i$ from his view $\mathrm{View}_{P_i}(\mathbf{a}_1, \ldots, \mathbf{a}_n)$. More precisely, we have the following definition.

Definition 1. *We say that the protocol Π realizes ϕ with* correctness *if for any input $(\mathbf{a}_1, \ldots, \mathbf{a}_n)$, it holds*[7] *that* $\Pr[\,\phi(\mathbf{a}_1, \ldots, \mathbf{a}_n) \neq$ *output of Π for P_i* $] = 0$ *for all $i \in [n]$. Let $\mathcal{A}$ a subset of at most $n-1$ players, the protocol Π_f realizes ϕ with* privacy *against $\mathcal{A}$ if it is correct and there exists a PPT algorithm* Sim *such that* $(\mathrm{View}_{P_i}(\mathbf{a}_1, \ldots, \mathbf{a}_n))_{P_i \in \mathcal{A}}$ *and* $\mathrm{Sim}((\mathbf{a}_i)_{P_i \in \mathcal{A}}, \phi(\mathbf{a}_1, \ldots, \mathbf{a}_n))$ *are computationally indistinguishable for all inputs.*

Protocol Π

- Parties: CSP and MLE with no input
  DO_i with input $\mathcal{D}_i$ for $i = 1, \ldots, m$.
- Output: each party gets $\tilde{\mathbf{w}}^*$
- Public parameters: number of features d, total numbers of data points n, regression parameters $s, \lambda, t, \mathbf{w}_1$.

Phase 1: MLE, CSP and $\mathrm{DO}_1, \ldots, \mathrm{DO}_m$ jointly run Π^1_{hor} or Π^1_{ver};

Phase 2:
- MLE and CSP jointly run Π^2_{ridge} or Π^2_{lasso};
- MLE sends to the other parties $\tilde{\mathbf{w}}^*$.

Fig. 6. The protocol Π implements our system.

Theorem 1. *Let Π be the protocol described in Figure 6 and ϕ be the functionality computing the linear regression model $\mathbf{w}^*$ from the data represented by $X \in \mathbb{Z}_q^{n \times d}$ and $\mathbf{y} \in \mathbb{Z}_q^n$ (see Table 1). Let $D \subseteq \{1, \ldots, m\}$. Then, Π realizes ϕ with correctness and privacy against the adversaries $\mathcal{A}_1 = \{\mathrm{MLE}\} \cup \{\mathrm{DO}_i \mid i \in D\}$ and $\mathcal{A}_2 = \{\mathrm{CSP}\} \cup \{\mathrm{DO}_i \mid i \in D\}$.*

Proof. Correctness can be easily verified from the homomorphic properties of the underlying encryption scheme, and Lemmas 3 and 2. To prove privacy we construct two simulators Sim_1 and Sim_2 which simulate the view of the parties

[7] The probability is over the choice of the random tapes $\mathbf{k}_i$.

	Input	Parameters	Algorithm	Output
RIDGE	$X, \mathbf{y}$	λ	Compute $\mathbf{w} = (A + \lambda I)^{-1}(\mathbf{b})$	$\mathbf{w}$
LASSO	$X, \mathbf{y}$	s, t, $\mathbf{w}_1$	For $j = 1, \ldots, t-1$ $\begin{cases} \mathbf{v}_j \in \text{argmin}_{\lVert \mathbf{v} \rVert_1 \leq s}\{(A\mathbf{w}_j - \mathbf{b})^\top \mathbf{v}\} \\ \mathbf{w}_{j+1} = (1-\gamma_j)\mathbf{w}_j - \gamma_j(\mathbf{v}_j) \end{cases}$	$\mathbf{w}_t$

Table 1. Compact description of the functionality we consider in order to compute a regression model $\mathbf{w}^*$ from the training set represented by the feature matrix X and the outcome vector $\mathbf{y}$. Recall that $A = X^\top X$ and $\mathbf{b} = X^\top \mathbf{y}$.

in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.

$\text{Sim}_1(\{\mathcal{D}_i\}_{i \in D}, \mathbf{w}^*)$:

1. Run $\mathsf{Gen}(\kappa) \rightarrow (\mathbf{pk}, \mathbf{sk})$;
2. For any $i \in \{1, \ldots, m\} \setminus D$, sample $\mathcal{D}_i$;
3. Sample R and $\mathbf{r}$ as in the protocol;
4. Compute $\tilde{\mathbf{w}} = R^{-1}(\mathbf{w}^* + \mathbf{r})$;
5. Output $(\mathbf{pk}, \{\mathcal{D}_i\}_{i \in D}, enc, \tilde{\mathbf{w}}, \mathbf{w}^*)$ where enc contains the encryptions of the values $\mathbf{x}_i, y_i$ for all i if we are in the vertically-portioned setting, or the encryptions of the values $\mathbf{x}_i\mathbf{x}_i^\top, y_i\mathbf{x}_i$ for all i if we are in the horizontally-portioned setting.

It follows from the semantic security of the encryption scheme that the simulation output has the same distribution of the views of the corrupted parties in $\mathcal{A}_2$ in the protocol Π.

$\text{Sim}_2(\{\mathcal{D}_i\}_{i \in D}, \mathbf{w}^*)$:

1. Run $\mathsf{Gen}(\kappa) \rightarrow (\mathbf{pk}, \mathbf{sk})$;
2. Sample R and $\mathbf{r}$ as in the protocol;
3. Compute $\mathsf{Enc}_{\mathbf{pk}}(R)$ and $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{r})$;
4. Output $((\mathbf{pk}, \mathbf{sk}), \{\mathcal{D}_i\}_{i \in D}, \mathsf{Enc}_{\mathbf{pk}}(R), \mathsf{Enc}_{\mathbf{pk}}(\mathbf{r}), \mathbf{w}^*)$

It follows from Lemma 1 that the simulation output has the same distribution of the views of the corrupted parties in $\mathcal{A}_1$ in the protocol Π.

Active Security. The protocol Π guarantees privacy when all the parties follow the instructions given in the protocol (passive security). Here we briefly discuss the security of Π in the case when the CSP or the MLE are corrupted and arbitrarily deviate from the protocol. We still assume that they do not collude.

If the CSP is corrupted, during Phase 2 it can send to the MLE a faulty $\tilde{\mathbf{w}}$ causing the computation of a wrong model $\tilde{\mathbf{w}}^*$. In this case, if we are in the horizontally-partitioned setting, the each data-owner DO_i can verify on their local data the received model $\tilde{\mathbf{w}}^*$ and catch the cheating CSP. That is,

DO_i checks that $(\tilde{\mathbf{w}}^*)^\top \mathbf{x}_i - y_i \in [-u, u]$ for a small u chosen by the parties. If all data-owners do not complain, the received model is valid. If we are in the vertically-partitioned setting, then the data-owners can jointly run an m-party MPC protocol to securely compute the sum of m inputs and again check that $(\tilde{\mathbf{w}}^*)^\top \mathbf{x}_i - y_i \in [-u, u]$ for all i.

If the MLE is corrupted, then it can decide to ignore o replace some of the ciphertexts received during Phase 1. This may be reveal extra information about some of the private datasets $\mathcal{D}_1, \ldots, \mathcal{D}_m$. A solution to avoid this threat in the horizontally-partitioned setting is proposed in [NWI+13]. They use one-time MACs, Pedersen commitments and standard zero-knowledge proofs. Extending this solution to the vertically-partitioned using multiplicatively homomorphic commitments is left for future work.

Discussion. Notice that, as the protocol Π works with arithmetic on a finite ring, it computes an approximation of the original linear regression functionality (the one computing a linear regression model directly from real-valued data). In [FIM+06], Feigenbaum et al. show that the output of an approximation may reveal more information than the output of the original function. For this reason, they define the notion of security approximation and give a protocol to make an approximation satisfy such definition. A detailed study of our system in the framework proposed by Feigenbaum et al. is left for future work.

References

[AHPW15] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Fast and secure linear regression and biometric authentication with security update. Cryptology ePrint Archive, Report 2015/692, 2015.

[AP09] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, pages 75–86, 2009.

[AS00] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 439–450, 2000.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 325–341, 2005.

[Bub15] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

[C+09] International Warfarin Pharmacogenetics Consortium et al. Estimation of the warfarin dose with clinical and pharmacogenetic data. *N Engl J Med*, 2009(360):753–764, 2009.

[CDNN15] Martine De Cock, Rafael Dowsley, Anderson C. A. Nascimento, and Stacey C. Newman. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In *Proceedings of the 8th*

ACM Workshop on Artificial Intelligence and Security, AISec 2015, Denver, Colorado, USA, October 16, 2015, pages 3–14, 2015.

[DHC04] Wenliang Du, Yunghsiang S. Han, and Shigang Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004*, pages 222–233, 2004.

[FIM+06] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Secure multiparty computation of approximations. *ACM Trans. Algorithms*, 2(3):435–472, 2006.

[FT07] Michael P. Friedlander and Paul Tseng. Exact regularization of convex programs. *SIAM Journal on Optimization*, 18(4):1326–1350, 2007.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, pages 506–522, 2010.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[GSB+16] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. Cryptology ePrint Archive, Report 2016/892, 2016.

[HFN11] Rob Hall, Stephen E Fienberg, and Yuval Nardi. Secure multiple linear regression based on homomorphic encryption. *Journal of Official Statistics*, 27(4):669, 2011.

[KLSR04] Alan F. Karr, Xiaodong Lin, Ashish P. Sanil, and Jerome P. Reiter. Regression on distributed databases via secure multi-party computation. In *Proceedings of the 2004 Annual National Conference on Digital Government Research*, pages 108:1–108:2. Digital Government Society of North America, 2004.

[KLSR05] Alan F Karr, Xiaodong Lin, Ashish P Sanil, and Jerome P Reiter. Secure regression on distributed databases. *Journal of Computational and Graphical Statistics*, 14(2):263–279, 2005.

[KLSR09] Alan F Karr, Xiaodong Lin, Ashish P Sanil, and Jerome P Reiter. Privacy-preserving analysis of vertically partitioned data using secure matrix products. *Journal of Official Statistics*, 25(1):125, 2009.

[LP00] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 36–54, 2000.

[NWI+13] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 334–348, 2013.

[SKLR04] Ashish P. Sanil, Alan F. Karr, Xiaodong Lin, and Jerome P. Reiter. Privacy preserving regression modelling via distributed computation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 677–682. ACM, 2004.

[VZ12] Salil Vadhan and Colin Jia Zheng. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In *Proceedings of the 44th symposium on Theory of Computing*, pages 817–836. ACM, 2012.